IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received September 22, 2020, accepted October 6, 2020, date of publication October 12, 2020, date of current version October 27, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.3030297

Secure Edge Computing Management Based on Independent Microservices Providers for Gateway-Centric IoT Networks

WENQUAN JIN[1], RONGXU XU[2], TAEWAN YOU[3], YONG-GEUN HONG[3], AND DOHYEUN KIM[2]

[1]Big Data Research Center, Jeju National University, Jeju City 63243, South Korea
[2]Department of Computer Engineering, Jeju National University, Jeju City 63243, South Korea
[3]Electronics and Telecommunications Research Institute, Daejeon 34129, South Korea

Corresponding author: Dohyeun Kim (kimdh@jejunu.ac.kr)

This work was supported in part by the National Research Foundation of Korea (NRF) Grant funded by the Korean Government under Grant NRF-2019R1I1A1A01062456, and in part by the Energy Cloud Research and Development Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Science, ICT under Grant 2019M3F2A1073387.

ABSTRACT Edge computing is an emerging computing paradigm that distributes the computational capability to the edge of networks for enabling the computation near to the environment where the sensors and actuators are deployed. Therefore, from the network edge, heterogeneous solutions can be provided to the Internet based on sufficient computing ability. Nevertheless, computing and networking resources are constrained for devices in the network edge. Providing secure services from edge computing is a challenge based on constrained resources. In this paper, we propose a secure edge computing to provide management of device, data, user and additional services based on deploying independent microservices providers with a security gateway on an edge gateway. The edge gateway is the hub of a local network where multiple IoT devices are deployed to interact with the physical environment for sensing and actuating. The gateway provides the management functionalities through microservices based on multiple independent server modules. Each gateway-centric local network has a standalone management service based on the gateway. For providing secure edge computing services through the edge gateway, a security gateway is deployed on the proposed edge gateway to provide Representational State Transfer Application Programming Interfaces to expose the security services to the Internet instead of microservices from management modules. Moreover, a client support gateway is deployed in the edge gateway to provide services of User Interface and access forwarding based on web sessions to support user authentication and authorization with the security gateway. Based on the proposed edge gateway including client support and security gateway, IoT clients and IoT devices are enabled to communicate for providing secure edge services of access and visualization to users.

INDEX TERMS Internet of Things, edge computing, edge gateway, device, management, security, microservices, API gateway.

I. INTRODUCTION

The Internet of Things (IoT) is comprised of heterogeneous Internet-connected devices that cover multiple Internet layers to provide ubiquitous and intelligent services. The seamless IoT services are aimed to make people's lifestyle to be safe and comfortable based on IoT devices including sensors and actuators in private and public spaces. Most of IoT devices are developed for constrained environments using the small size of equipment to provide simple services based on limited resources [1]. Cloud computing is effectively scalable and easily accessible for the limitations of IoT devices such as low power, heterogeneous machine type of connectivity and lack of computing resources [2]. A gateway-centric local network can be a part of an IoT network to provide services in the local space. Local networks can be deployed on the edge of the IoT network to provide IoT services to Internet users based on IoT devices which are also deployed in the local network. Therefore, users on the Internet, who access IoT services based on gateways from multiple local networks.

The associate editor coordinating the review of this manuscript and approving it for publication was Tawfik Al-Hadhrami.

For the concept of gateway-centric local networks, the paradigm of edge computing provides a computing concept on the edge of networks. Edge computing has the potential to address the concerns of network communication cost, power supply, and data safety and privacy in local networks [3]. Based on the concept of edge computing, that enables IoT devices to take full advantage of IoT and cover limitations such as lack of physical resources and privacy.

The edge computing enables the computation to be performed at the edge of the network based on the interactions with other network elements including cloud servers and IoT devices. In IoT networks, data is increasingly generated at the edge of the networks and efficient solutions are desired such as micro datacenter [4], [5], cloudlet [6], and fog computing [7]. As a middle layer element between IoT devices and cloud servers, the edge computing deploys the devices to interact with servers and clients. According to the architectural concepts of edge computing, which aims to satisfy key requirements such as scalability, multi-tenancy, security, privacy, and flexibility [8]. The edge gateway is a device that provides functions of edge computing and interacts with the IoT devices as well as the cloud [9]–[11]. Based on the sufficient computational ability, the edge gateway is enabled to support management services such as monitoring the environment and devices, registering the information of devices for discovery, and forwarding messages between different protocols and networks [12]–[15]. Using these services, clients get the sensing data of the environment and control the actuators to change the environment where the devices are deployed with the edge gateway.

In IoT networks, the management functionalities are required for the good interoperability of devices to enable the IoT services to be visible and accessible on the Internet [16]. Management of the device is used for reducing time to configure and manage the IoT devices through network communications remotely [17]. The devices are represented in the Internet through web resources to be managed as cyber objects. Using IoT device management, heterogeneous devices can be categorized to represent the devices via digital information and integrate data to the Internet [18], [19]. For managing various IoT devices in various local networks, a unifying management system is required which supports consistent service interfaces, communication protocols and data formats. Standard technologies can be used for developing the management functions to unify the transmission data formats, interaction interfaces, communication technologies. Securing data in IoT networks is very important to protect privacy [20], [21]. Based on the IoT devices, the data is generated in real-time and anywhere including private spaces. For securing the services from an edge gateway with the local network where the edge gateway is deployed, a gateway can be deployed in the edge gateway to filter the requests from the out of local network [22]. Therefore, the security gateway enables to allow authenticated and authorized web clients to access the local network where the edge gateway deployed.

For supporting the secure management service of IoT devices in private networks based on the edge gateway, in this paper, we present an edge computing management based on secure microservices to provide management of device, data, user, configuration and other additional functions. The proposed edge gateway is deployed in the entry of the network edge and includes all functions in the standalone server to provide the private services for the private users. The microservices enables the integration of the extendable applications into the server. The gateway provides the management of IoT devices to enable devices and data access from a network edge. Based on the multiple server modules, the gateway provides flexible microservices for management and bridging communications between web clients to IoT devices. For securing services from the local network, a security gateway is deployed in the edge gateway to provide security functions. The security gateway provides management of authentication and authorization to secure the multiple microservices providers for the private environment. In the implementation, the modules of management are configured based on a standard that is EdgeX framework from EdgeX foundry. The EdgeX Foundry is a standardization organization that published an edge computing framework trough the Linux Foundation and Dell to support a set of functions for IoT and edge computing [23]. The security gateway exposes secure services to the Internet for filtering access based on authentication and authorization services. The security service is supported by Kong [24] that is integrated into the edge gateway. For delivering the data from the edge to a web client, the client support gateway is deployed in the edge gateway to provide services of User Interface (UI) and access forwarding.

The rest of the paper is structured as follows. Section II introduces the related works regarding edge computing and its management with security solutions and microservices. Section III presents the design of edge computing management based on secure microservices including IoT system architecture, proposed security edge computing management, and detail functional architectures. Section IV presents the implementation details and results for the proposed secure-microservices-based edge computing management. Section V presents the performance evaluation based on the implementations for accessing the IoT device and requesting the IoT data. Section VII presents a comparison with the existing solutions for emphasizing the significance of the proposed secure edge computing management. Finally, we conclude this paper and introduce our future directions in Section VII.

II. RELATED WORKS

Standard frameworks can be used for developing the proposed edge computing management including transmission data formats, interaction interfaces, communication technologies. For developing edge computing efficiently, several initiatives are proposed. Cloudlet [25] is consists of

computers and abundant resources to provide a trusted, resource-rich computer or cluster of computers which are well-connected to the internet and available for use by a nearby mobile device [26]. Using the architecture of cloudlets, M. Satyanarayanan *et al.* proposed an open ecosystem for mobile-cloud convergence [27]. Mobile edge computing (MEC) is proposed by the European Telecommunications Standards Institute (ETSI) to provide a data service and cloud computing environment at the edge of a network [28]–[31]. The main technologies of MEC are computation offloading and mobility management which enable distributed mobile cloud computing over 5G heterogeneous networks [32]. Tseng *et al.* proposed Industrial Internet Consortium (IIC) that considers the enterprise layer as an important element in which specific processes and applications in the cloud are executed such as decision support systems, operation management, big data analytics, model training, predictions and business analytics [33]. Sittón-Candanedo *et al.* proposed Global Edge that includes a level of security that starts at the base layer by encrypting sensor-generated data which are further processed at the Edge layer based on blockchain technology [34]. Fog computing is a distributed cloud computing platform at the edge of a network to provide computation, storage and management service [35]–[37]. CloudPath [38] is an edge computing system that is consists of a set of short-cycle and stateless functions to support a path computing architecture. EdgeX framework is proposed for IoT edge computing to support connection with heterogeneous sensors and devices based on different protocols, and management of devices and data [39]. EdgeX framework is consists of multiple server modules that provide microservices to allows services to scale up and down based on the capability of the edge gateway.

The management includes several domains such as application, network and system for configurations and monitoring of the resource in IoT networks [40]. In IoT networks, devices have been deployed to provide sensing and actuating services in various industries such as manufacturing, supply chains, energy, healthcare, and the automotive industry [41]–[43]. In edge computing, the local network is operated by the edge gateway that interacts with IoT devices. The management functions can be deployed in the edge gateway to provide services IoT devices as well as other elements on the Internet. For IoT management including management of devices and data, various standards have been proposed. The Open Mobile Alliance (OMA) is an international standard organization that proposed a device management architecture for managing devices using request and response transaction models through client and server [44], [45]. The Lightweight M2M (LWM2M) is the implementation of OMA that supports communications of a device based on CoAP that is used for constrained devices [46]. The Open Connectivity Foundation (OCF) is a standard framework for implementing IoT devices on various software environments including mobile phones and wearable devices based on Android, iOS, Linux, Tizen, etc [47], [48]. For communications between IoT devices, the OCF adopts CoAP as the default protocol to provide services based on Representational State Transfer (REST) Application Programming Interface (API) REST API. The IoTivity is developed to support developers to include OCF functions in applications for implementing the role of client and server [49], [50]. The specification of OCF provides guidelines for the functionality of messaging, discovery, monitoring, and maintenance based on the fundamental communication ability.

IoT platforms are used for building an IoT system to provide a set of functions including device communication, management, storage and service. Charilaos Akasiadis *et al.* proposed an IoT platform that supports multiple protocols based on protocol implementations in the platform server [51]. Mainly, the most commonly used IoT protocols including MQTT, AMQP, WebSockets, CoAP, and HTTP are involved with oneM2M common service layer to provide IoT services. The oneM2M is an IoT specification that provides scalable and interoperable IoT standards for the communication of devices and services [52]. According to the main goal of oneM2M, that pursues to build a single horizontal service platform for all functions in various industrial domains. Nevertheless, the architecture of oneM2M is too heavy to deployed on the local environment for bridging the Internet and private space. Zamora-Izquierdo *et al.* proposed a flexible IoT platform based on exchangeable low-cost hardware and supported by a three-tier open source software platform at local, edge and cloud planes [53]. However, security is not considered to provide authentication and authorization functions for service clients. Mijić *et al.* presented a scenario to qualify an IoT platform in the perspective of performance, scalability, availability, security, connectivity, and deployability [54]. For these quality attributes, the proposed edge gateway can be satisfied to support flexible maintenance, sufficient performance and secure communication based on the EdgeX framework with security and client support gateways.

With the increasing scale of the IoT systems, development of IoT application to be flexibility, lightweight and loose coupling is important to enable extensibility and maintainability, and the microservice architecture is a key to meet the demand [55]–[59]. Benayache *et al.* [60] proposed a middleware in IoT networks for enabling service interconnectivity and availability over heterogeneous networks based on accessing different APIs and communication protocols. Fernandez *et al.* [61] proposed an orchestration of IoT slices through providing microservices from edge and cloud environment. Moreover, several smart solutions are developed based on microservice architecture through deploying multiple server entities for providing sufficient scalability and performance [62]–[64]. However, providing secure IoT services from the distributed service providers is difficult due to management of authentication and authorization [65], [66]. For applying security to the edge gateway for securing microservice, an API gateway can be the entry to filter the requests through forwarding messages between microservices and

web clients [65]. For the EdgeX framework, the API gateway integrates JWT and OAuth2.0 to support authentication and authorization for providing access rights to clients [67]. For configuring an API gateway, Kong is an orchestration to provide a flexible middleware between clients and microservices using API [68]. Kong is available as open source project to support high performance and extensibility.

III. DESIGN OF EDGE COMPUTING MANAGEMENT BASED ON SECURE MICROSERVICES

A. SYSTEM ARCHITECTURE FOR GATEWAY-BASED IoT NETWORK

For the proposed gateway-based IoT network including multiple local networks with edge gateways, we present an IoT architecture to introduce interactions of elements in the IoT network and functional blocks in the edge gateway. In the overall system architecture, local networks are connected to the Internet through routers that provide network services to edge gateways and IoT devices. As shown in Figure 1, the proposed gateway-based IoT architecture includes IoT clients, edge gateways, and IoT devices to provide services to users.

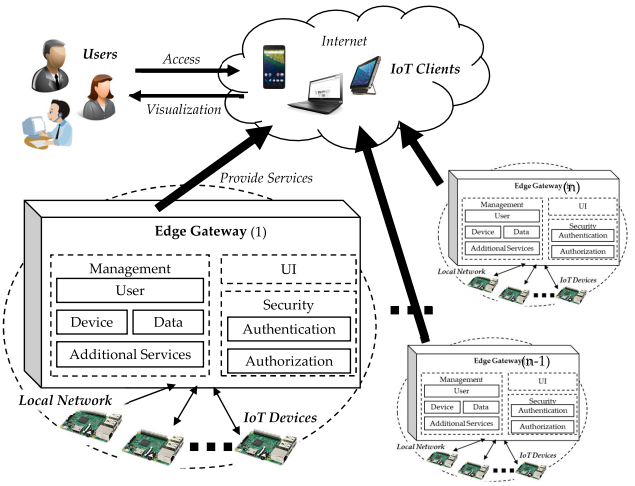


FIGURE 1. Proposed gateway-based IoT architecture.

The edge gateway is deployed in a local network with IoT devices to provide IoT services such as sensing and actuating for collecting data from the environment and changing the parameters in the environment. The gateway-based IoT architecture is comprised of the local networks and IoT clients. in the Internet, the IoT clients access the local networks using the services that are provided by the edge gateways. On the Internet, IoT clients are used by users to access the services of edge gateways. The information from the edge gateways through services, which are interpreted in the IoT clients and displayed to users. Based on the edge gateway, the proposed IoT system provides services including management of user, device, data and additional services. The IoT environment based on an edge gateway, that is a standalone system to provide services. The management functionality has been proposed by IoT frameworks such as oneM2M [52], OCF [47] and LWM2M [46]. However, the extendable nature of microservices-based development style is the key to support the development of lightweight and various functions in the edge computing. Based on the EdgeX, the edge gateway is developed by deploying various microservices server modules to provide multiple functions including security and UIs.

Moreover, the local network can be supported for a private space where the IoT devices are connected to the local network. As a fundamental system for an IoT environment, the gateway-based IoT system is used for a private network environment such as smart home, smart factory, hospital, and other smart spaces where the IoT devices are connected to the private network except the Internet. Therefore, the IoT clients cannot access the services of IoT devices directly. The edge gateway bridges IoT devices and the Internet to enable transparent accesses for IoT clients. In a local network, an edge gateway provides services to IoT clients for the management and access of context. As a web server, the edge gateway exposes web APIs to the Internet based on microservices. In the internal of the edge gateway, the functions are implemented based on calling other microservices.

The edge gateway is deployed in an environment to provide services according to the environment where the IoT devices are deployed to collect data and update status of the environment. A local network provides the communication for the environment to enable access of the services through the edge gateway. In the local network, the management services are used for managing the device that are deployed in the network edge. From the edge gateway, the UIs are provided to the access the management services. With the UIs the authentication and authorization services are provided through the security functions.

Figure 2 presents a hierarchical architecture of functional components that are divided into three layers including the layer of client, gateway, and device. The client layer includes user interfaces of management, visualization and user authentication for interacting with the gateway layer. The gateway layer includes the edge gateway that bridges the client and device layer to enable accesses of IoT services in local networks. In the edge gateway, multiple layers are included based on various functional blocks. The client support gateway is a server that is a part of the edge gateway to provide interface services and user support modules such as login, registration, and user management through interacting with internal microservices.

The security gateway is used for securing microservices of the edge gateway that includes functions of authorization and authentication. The services of core and additional provide management and access functionalities based on microservices through the security gateway to services clients. The module of metadata, data, command, and configuration are standalone microservice servers that are deployed in the edge gateway. Metadata module provides management of information about the devices to enable communications with devices. Data module provides a centralized persistence

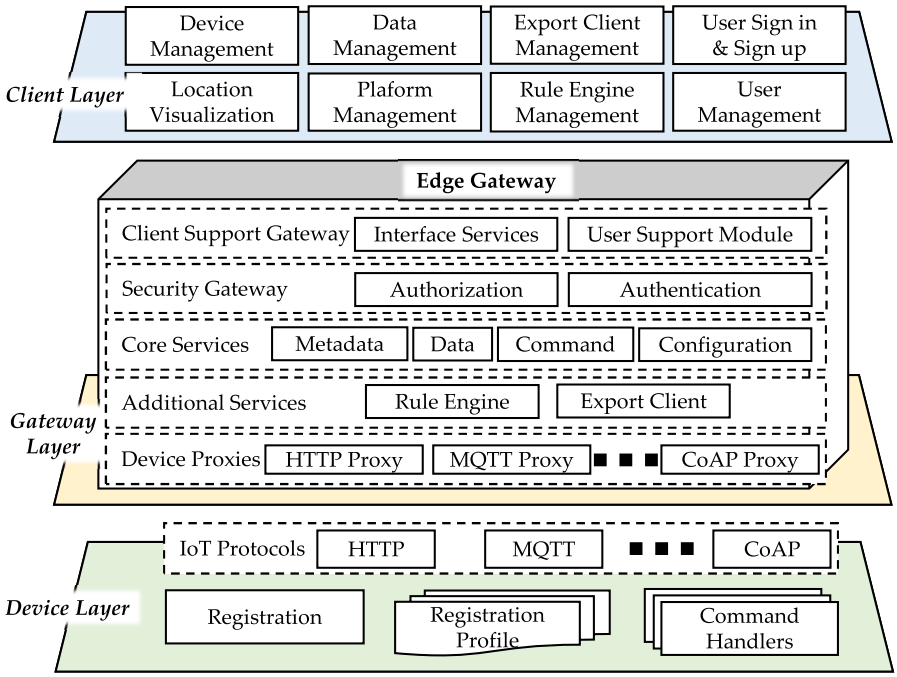


FIGURE 2. Hierarchical architecture of functional components.

functionality for storing data readings that are collected by devices.

For securing REST APIs, OAuth and JWT are the major solutions that are based on tokens to identify granted requests [69]–[71]. As well as in IoT, most of the protocols are designed for constrained environment and bi-direction-request based on light-weight and subscribe-publish architectures. The security gateway is a part of the edge gateway that grants tokens to clients for enabling the requests to microservices.

B. SECURE MICROSERVICES OF EDGE COMPUTING MANAGEMENT

For securing microservices-based edge computing management functions on the proposed edge gateway, the collaboration of API gateway and session mechanism is deployed in the edge gateway. The API gateway is a module of security gateway that filters all incoming requests for securing the microservices. Moreover, the client support gateway provides UIs to web clients through session-based authentication.

Figure 3 shows the proposed edge computing network architecture based on secure microservices. The security gateway is included in the edge gateway to provide security services for securing microservices based on authentication and authorization. The agent provides essential functions to use API gateway microservice based on REST API to clients. The agent is comprised of a controller, volume, config loader, handler, and the requestor. Controller receive request from the client then invokes a related function to the response. Volume includes toml config file which is the manifest of the microservices API gateway. The public key is used to generate the JWT certificate then verify JWTs. Config loader parses specific configurations from toml file then created related instances to provides information of microservices. The handler provides business logic to generate the consumer-related request and service-related requests such as additional service, route, consumer and enable JWT plugins to generate a certificate of JWT. The requestor is responsible send requests to API gateway through admin API.

For providing user authentication, the client support gateway provides the login UI to the client. Through the client, the edge gateway authenticates the user and allows to access the microservices. Once a user is successfully authenticated, then the edge gateway authorizes the client to request the microservices through the API gateway based on the assigned token. The client support gateway is a web server that provides the session mechanism to record authorized clients. Therefore, the authorized clients are stored in the client support gateway with the assigned tokens.

Figure 4 shows the authentication and authorization process based on the proposed edge gateway with the web client, client support gateway, and the security gateway. Web client requests the service /signup to get the signup page and displays to a user. The user inputs registration information including ID, password, user's name and mode on the signup page, and sends it to the service /addUser through the web client. The client support gateway requests to the security gateway to create an authentication object and get a token for the user. Then, the client support gateway sends the created

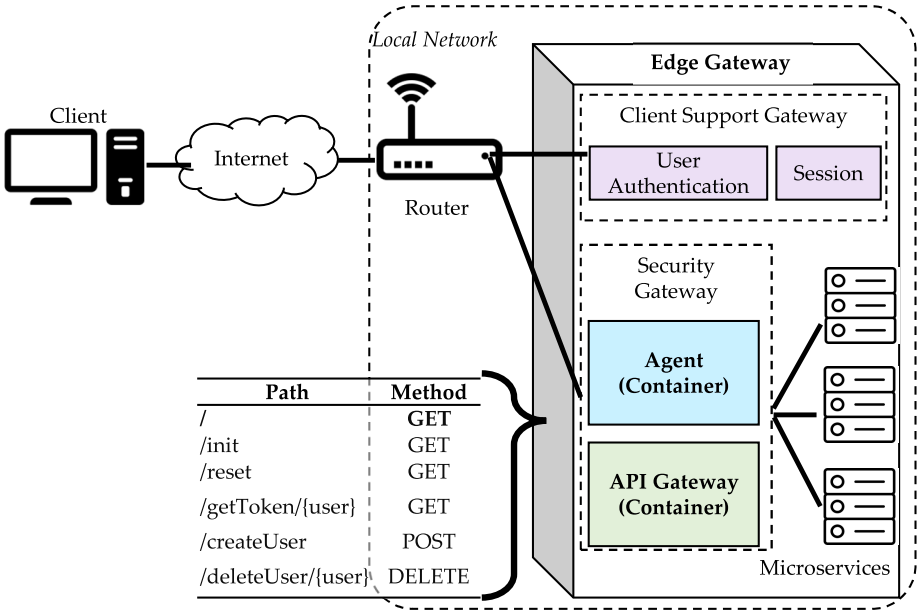


FIGURE 3. Proposed edge computing network architecture based on secure microservices.

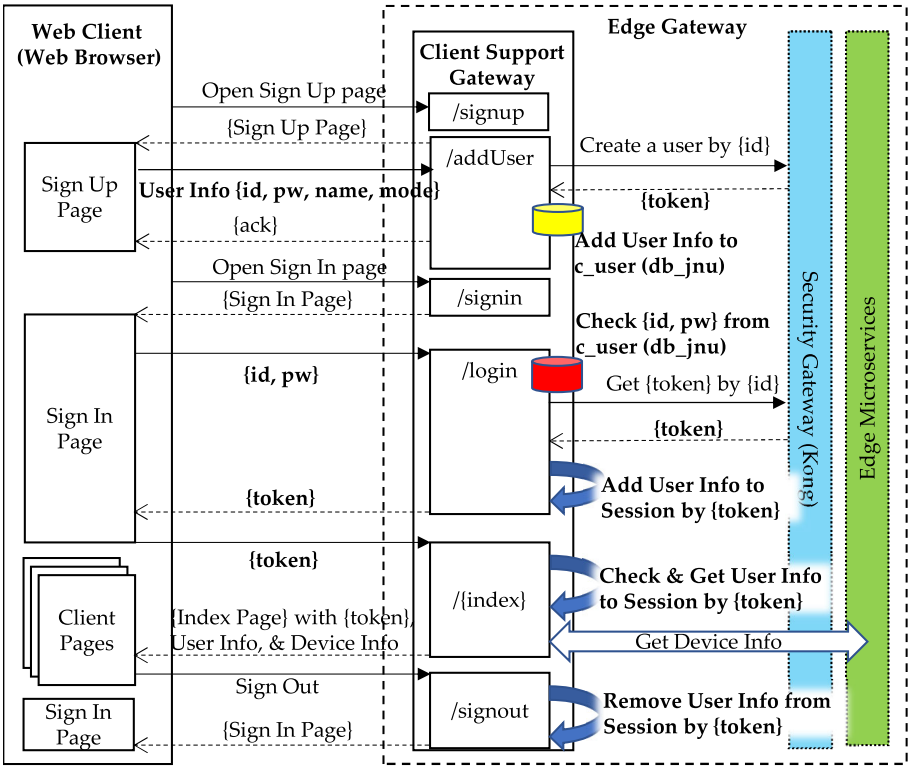


FIGURE 4. Authentication and authorization process based on the proposed edge gateway.

token with ID, password, name, and mode to the MongoDB using edge microservices. The web client requests the service /signin to get the sign in page and displays to the user. The user inputs the ID and password to the sign in page, and the page sends values to the client support gateway for verifying the user based on the MongoDB. EdgeX framework includes a MongoDB based on the Docker container. The DB is a document type that includes collection c_user to save user information which is registered by users. The authentication is successful through the login, then the web client has the

token that is used for accessing the microservices from the edge gateway. By the client support gateway, the session also is used for authorizing a client for filtering the request to the edge gateway. By accessing the service /signout, the client support gateway removes the session by the token to end the security service to the user.

For securing the edge computing services the collaboration of client support gateway and security gateway support the functions of authorization and authentication to clients. For authorization, users need to register the information to the edge gateway through the provided form. The registered information is used for the authentication of the user through verifying ID and password. The verified user can be assigned a token from the security gateway for the authorization process. Using the token, client support gateway registers a session for the user to keep the token and user information in the edge gateway temporarily. Therefore, the edge gateway removes the service access right through terminating the client or sign out.

C. IoT MANAGEMENT BASED ON EDGE MICROSERVICES

The proposed edge gateway provides management services including management of device, data, and supporting services based on microservice server modules. The management services are delivered to a web client through the client support gateway and presented to a user by UIs. In this section, we present the interactions between the web client and edge gateway to depict functions for managing devices, data, and supporting services.

Figure 5 shows functional architecture for managing IoT devices using the proposed edge gateway. For providing the IoT device management service to the web client, the microservices modules of edgex-mongo, edge-core-metadata, and edgex-core-command are deployed to provide services. The web client requests the device index page from the service /device with the device information including location list and number of registered devices. Device index page is delivered from client support proxy that provides web services to the clients which are web browsers in the client devices. The service /device are provided by client support proxy once a user is logged in through the user login page. Web client requests the device list page from the service /devicelist using the device index page. The service /devicelist returns the device list page, with the device list obtained by requesting metadata of EdgeX framework. Through device index page, a user can access device list page which is delivered by service /devicelist with a device list. The items of the device list provide functions to access a page that provides detail information of device. Web client requests the device detail page from the service /devicedetail using the device list page. The service /devicedetail returns device detail page with the device detail information that is obtained by requesting metadata of EdgeX framework. For getting the detail information of a device, client support proxy accesses metadata of EdgeX framework that provides the service for responding a device information by the device ID. Web client requests the device command page from the service /devicecommand using the device detail page. The service /devicecommand returns device command page with the device detail information that is obtained by requesting metadata of EdgeX framework for accessing the device. Through the retrieved device information, web client requests to the service /sendCommand with the device ID, command ID, request method type and parameters.

Figure 6 shows functional architecture for managing IoT data that is collected from the network edge using IoT devices. Based on requesting the microservices of the edgex-core-data module, the web client presents historical data, static data, and real-time data through interacting with EdgeX framework. The proposed edge gateway is deployed in the entry of a network edge where IoT devices are also deployed to provide sensing and actuating services. Through the edge gateway, the IoT devices can be accessed which means the IoT devices only interact with the edge gateway. The edge gateway includes a database to store the sensing and status data of IoT devices. The IoT device sends data through to

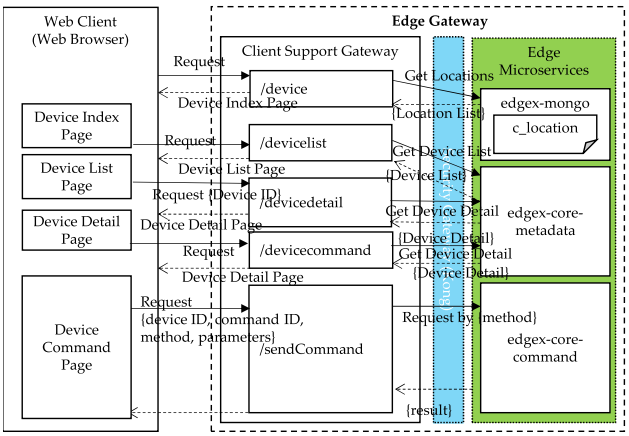


FIGURE 5. Functional architecture for IoT device management.

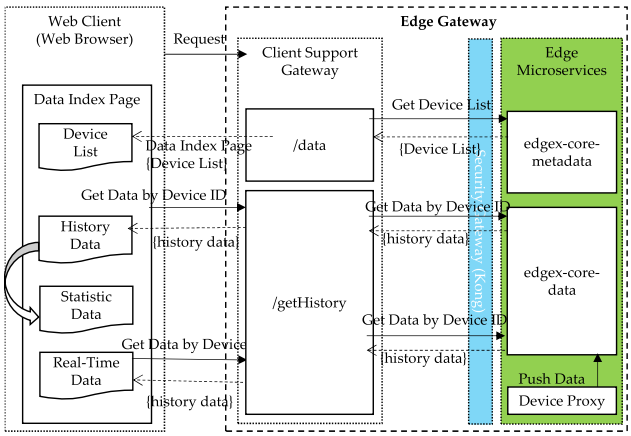


FIGURE 6. Functional architecture for IoT data management.

the edge gateway based on auto-event which is mentioned in EdgeX documentation. The auto-event enables the IoT device to sends data to the edge gateway periodically. Then, the stored data in the edge gateway can be updated by the latest value that is shown in the web client as real-time data.

The web client requests the stored data to present in the UIs. The history data is provided by accessing core data using a selected device ID. The statistical data is provided by calculating the history data in the web client. The real-time data is provided by accessing the device services of a selected device. The functions for presenting the history data and statistical data request the same data from service /getHistory. For presenting statistic data, web client stores the history data in the temporary variable and calculates the data to display as statistic charts.

Figure 7 shows functional architecture for managing supporting services. Using this function, the edge gateway provides the interface to accept the information of services. Therefore, the further implementation and deployment are required to provide completely functions. The support service management is used for managing additional services which are represented as services. Web client requests service /service to get the service index page with a service object list. Through the delete function of each item in the service object list, the web client requests to the service /delete to delete a selected service. Web client requests service /detail to get the service's detail page with the service detail information. The service detail information is retrieved by the selected service ID that is included in the service object list in the service index page. The service /detail accesses the supporting service to get the detail information of the selected service by ID and returns the information with the user interface. The web client requests service /form to get the service object form page and presents to the user. The user inputs the information of required fields and submits to the service /add for creating a new service object information. Using the submitted form data, the service /add generates a JSON format data for the new service object and sends it to the support service of EdgeX framework.

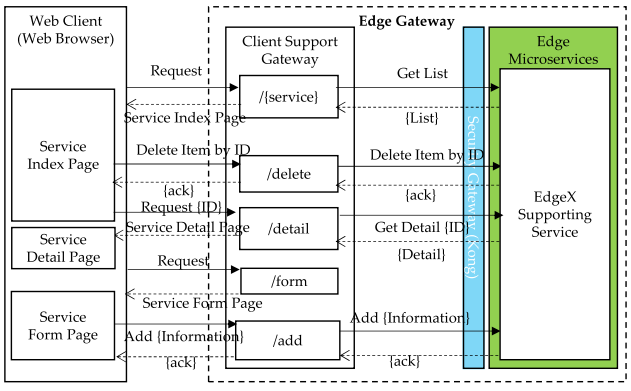


FIGURE 7. Functional architecture for supporting service management.

IV. IMPLEMENTATION OF EDGE COMPUTING MANAGEMENT BASED ON SECURE MICROSERVICES

For implementing the proposed edge computing management based on secure microservices, the elements including client support gateway, edge microservices, security gateway and device service are developed.

Figure 8 presents the implementation environment for secure edge computing management that is separated into the client, gateway, and device layers. The client support gateway is deployed in the edge gateway to provide UIs which are used by web clients. For implementing the server and client functions through the client support gateway, UI and server frameworks and libraries are used. JQuery and Bootstrap are used for implementing UIs and others are used for implementing server functions. UIs are provided by the client support gateway through services that are implemented in the server application. For providing services in the client support gateway, the Spring framework is used based on Java programing language with Windows operating system. Spring Boot is a web application development framework that is a part of the Spring framework. The Spring Boot is used for providing web services based on microservices. The client support gateway is built through Spring Boot that can be a standalone microservices provider to be deployed in the edge gateway easily. As optional solutions to develop microservices, embedded Apache server, Jetty, Flask, and Django also can be used for deployment in the proposed edge gateway.

For configuring the edge computing, the EdgeX framework Edinburgh 1.0.1 is used. The EdgeX is an edge computing framework that is comprised of various microservices providers to provide several management functions including device, network, data management. In the security gateway, Kong API gateway is used to provide security services, and additional security functions and device service are developed by Go 1.13.1 using GoLand on Ubuntu system.

For experimenting proposed edge computing based on the implementation of microservices, we deploy the microservice modules on an Ubuntu which can be installed on any device such as PC and RPi. The microservice modules are implemented based on Java for the client support gateway and Go for security gateway and EdgeX which can be deployed in any OS that can support these runtime environments. The EdgeX is a Go-based open-source implementation including multiple modules to provide edge computing services. For the additional services, Spring Boot based web servers also can be used for developing microservices. For the device, we use Android Things on RPi to implement an HTTP server application for providing services for sensing and actuating. Depending on the communication protocol of the IoT device, the device service needs to implement the specific protocol for the communication in the local network between the edge gateway and the IoT device. As shown in Figure 9, web client, edge gateway and IoT device are network entities in the experimental environment. The edge gateway is deployed in a

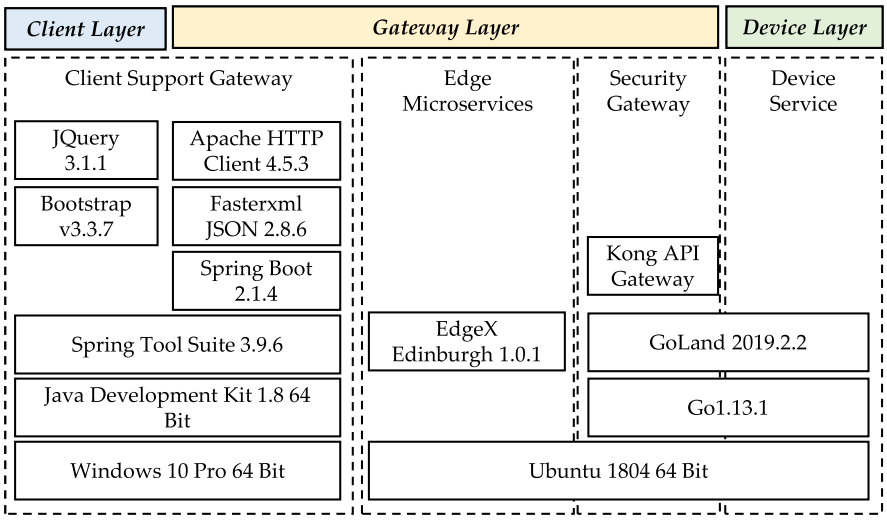


FIGURE 8. Implementation environment for secure edge computing management.

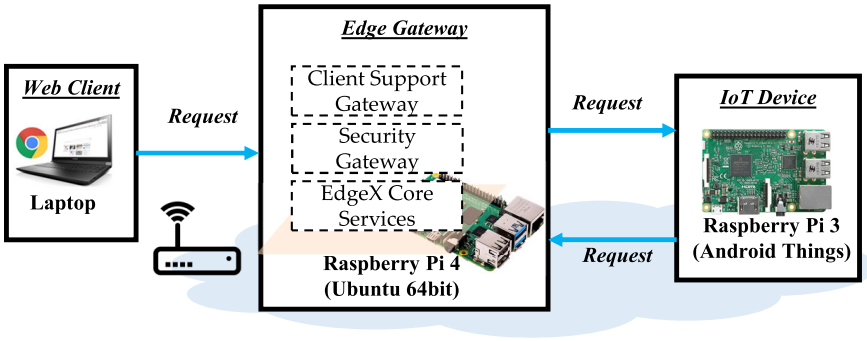


FIGURE 9. Experimental entities.

Raspberry Pi 4 with Ubuntu 64 bit and IoT device is deployed in a Raspberry Pi 3 with Android Things. Both devices are deployed in the same network, and the router only exposes the IP of edge gateway to the Internet. The Chrome web browser is used to access the edge gateway from a laptop through the Internet.

Figure 10 shows the implementation result of user registration and login in the web client. The user interface requires the user ID and password for verifying the login user and redirects to the page of device management. The user login interface for a general user that has a button which redirects to the normal device management. For an administrator, in the administrator login page, the button redirects to the device management which has an interface for managing users.

Figure 11 shows an implementation result of device management through the user interfaces of the web client to display a map with markers that present registered devices. Through this user interface, users can access the user management page and device list page. The link provides the user management interface to clients once a user clicks the link. The count of users presents the registered user numbers

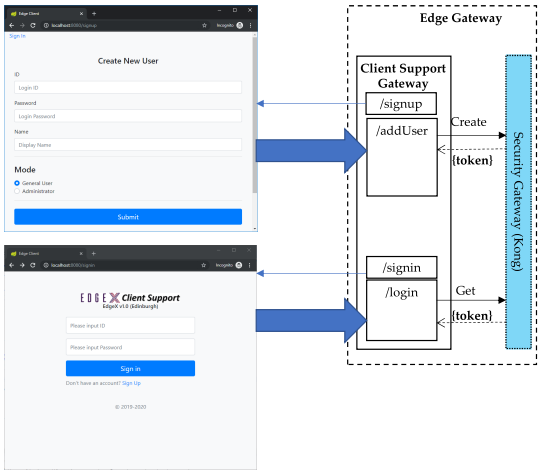


FIGURE 10. Implementation result of user registration and login.

including general users and administrators. The link devices provide the device list page to the client, in which users can retrieve the device list and access the device detail and control

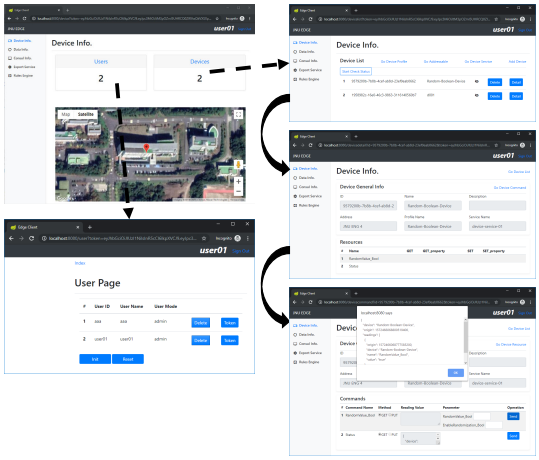


FIGURE 11. Implementation results of IoT device management.

devices. The count of devices presents the registered device numbers. A general user accesses this page using the general user login page, then the link users are not available on this page. The user interface displays the device list page that includes the device list and each item presents device ID, name and URI. An item of the device list provides the function of accessing the device detail page that presents detail information of the clicked items. The button detail provides a device detail page that presents a piece of detailed device information. The detailed device information is retrieved by the selected device ID. The device detail page includes device detail including the device's ID, name, description, address, profile name, service name and resource list. The information is retrieved by the selected device ID from Device List Page. The information delivered as a JSON format data provided from Metadata of EdgeX framework. The detailed information of a device includes properties of the device and a list of resources. Each resource in the list has two types of method that is used for accessing the resource to get the sensing data and control actuator. The device information is used for sending messages to resources of the device. For each resource of the device, a button is included for sending a request message to the resource using the selected method type. The response message is presented in this user interface using the alert window of the web client. The textbox of an item is in the command list that provides a radio button group to provide a method type for available method types of a resource. Therefore, for some resources, two method types are available, and for some resources, a single method type is available.

Figure 12 shows an implementation result of data management through the user interfaces of web clients. The device list is retrieved from metadata of EdgeX framework with buttons to display visual data in the bottom of the list. The user interface displays the device list page that includes the device list and each item presents device ID, name and URI. Each device item provides buttons to present historical data, statistic data and real-time data. The selected device may

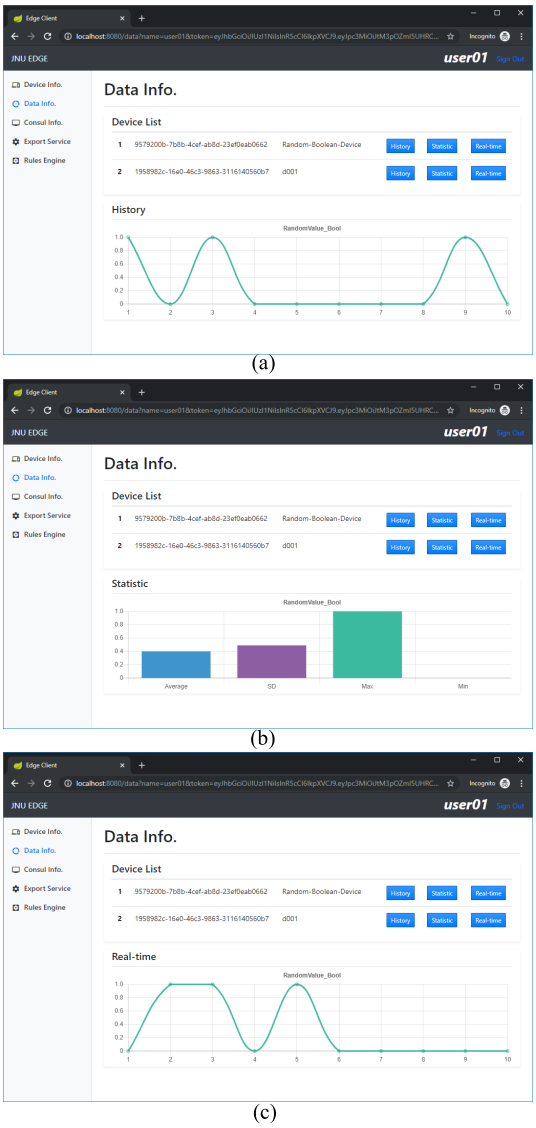
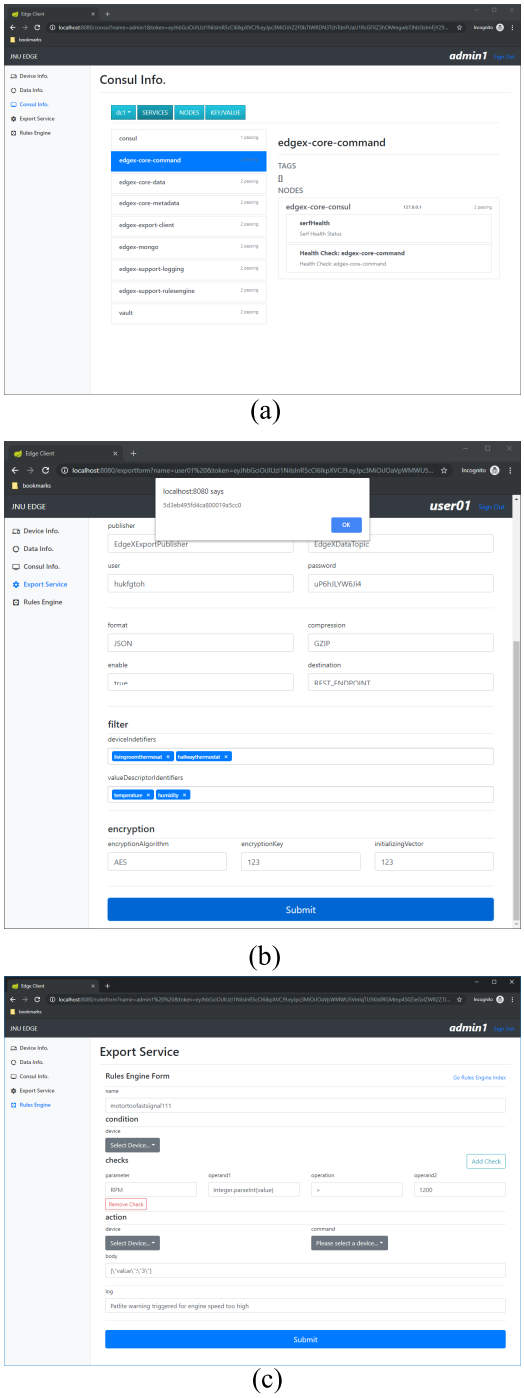


FIGURE 12. Implementation results of data management. (a) Historical data chart for selected device. (b) Summary of data for selected device. (d) Real-time data chart for selected device.

have multiple resources, and data of resources are sent to the EdgeX framework through event publishing. The data is stored in MongoDB of the EdgeX framework, and the web client requests the data from the EdgeX framework. The selected item of the list has three resources to provide IoT services. The first resource is an actuator that provides a control function to a fan. The second and third are sensors, therefore, the recorded data is available in the DB. For each resource of the selected device, the history data are requested and calculated to be statistic data. Once a device is selected for presenting the statistical data, the web client requests the EdgeX framework to get the history data and results in the statistical data for each resource. The first statistic bar presents the average of the retrieved data, second is the standard deviation, third is maximum, and the fourth

one is minimum. For presenting the data of each resource, the web client requests to the device and gets the real-time data. In order to access the selected device in real-time, web client requests EdgeX framework to get device information for sending the command to the device. The real-time data is collected by requesting to the resources of the selected device. In this process, web client requests to the device in periodic form, therefore, it consumes higher network resources.

Figure 13 shows the implementation results of configuration and additional management including management of configuration, export client, and rule engine. The configuration management is provided by Consul which is a tool to manage the microservices in the EdgeX framework. The client support gateway provides the management interfaces for interacting with the Consul services. In the UI, services are presented as a list that provides clickable items to display the details of a selected service. In the detail information of a selected service, the items link to the nodes. For each item in the service list, the service name and passing count are presented. For the detail information of service, tags and nodes of the service are presented. Export client is used for registering data subscribe in the EdgeX framework. For registering the export client based on UI in the web client, the client support gateway provides the UIs to web clients. For creating a new export client, the user interface displays the form page that includes fields for filling properties of export client that is used for creating a new export client profile. Once the fields are filled, through the Submit button to send the export client information to EdgeX framework. The submitted values of properties are sent to the client support gateway and included in a JSON format data. Rule engine enables the edge gateway to send a command to a specific device automatically based on the rule object. For creating a new rule using the UI of the web client, the form page includes fields for filling properties of the rule that are used for creating a new rule profile. Once the fields are filled, through the Submit button to send the rule to EdgeX framework. In this form, the drop boxes for the device in the condition and action areas, each shows a device list through requesting to Metadata.



(a)

(b)

(c)

FIGURE 13. **Implementation results of configuration and additional management. (a) Configuration management based on Consul. (b) Information form for the management of export client. (c) Information form for management of rule engine.**

V. PERFORMANCE EVALUATION

Edge computing deploys computation resources close to the network edge which reduces the latency of interactions for IoT elements. The edge gateway is deployed in the entry of the network edge that requires a sufficient computing ability for the private environment. Therefore, we collect the communication delays and edge gateway's memory usage to present the performance of proposed secure edge computing management.

For evaluating the implementation of the proposed edge gateway, the round-trip time (RTT) is collected from the interactions of the experimental environment. We ignore the network cost to test the applications including client support gateway, edge microservice and device service for evaluating secure edge services.

Figure 14 presents the interaction model for accessing IoT devices to get a result from an HTTP resource of the device. For experimenting with the performance, we collect the results based on two ways. In a first way, the request starts from the web client to the device service. The web client sends a request message to the client support gateway

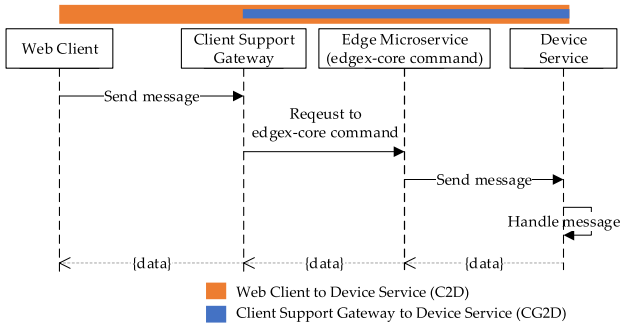


FIGURE 14. Interaction model for accessing IoT device.

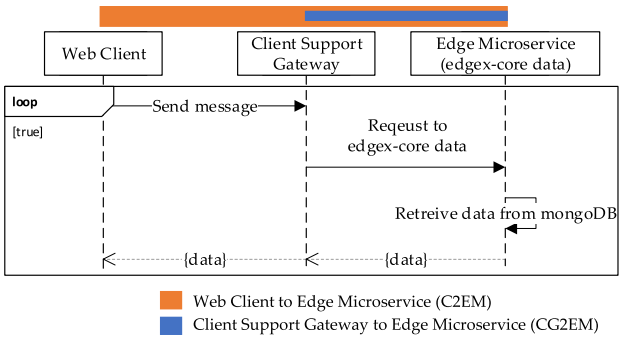


FIGURE 16. Interaction model for requesting IoT data.

and the gateway requests to the device service through the edge microservice. In this case, the web client deployed in another platform to request the edge gateway through HTTP. In a second way, the request starts from the client support gateway that is deployed in the edge gateway where also edge microservices are provided. Therefore, the results are generated in the gateway-centric local network.

Figure 15 shows the experimental results of time delays for accessing the IoT device for requesting from the web client to the device service (C2D) and requesting from the client support gateway to the device service (CG2D). The result depicts C2D takes more time than CG2D. The difference between C2D and CG2D can be expected to be the handling time by the client support gateway because of the experiment can ignore the network cost. The result is presented by CGD2 that can be used for evaluating the performance of the local network.

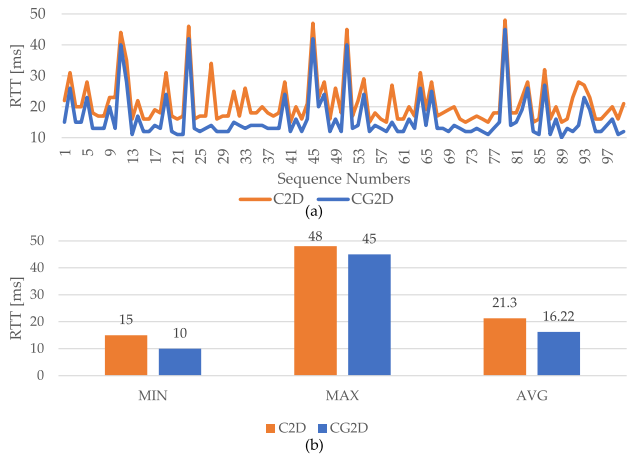


FIGURE 15. Experimental results of time delays for accessing IoT device.

Figure 16 presents the interaction model for requesting IoT data from the edge gateway that is deployed in a local network. In the first case, the web client requests the edge microservice (C2EM) that provides IoT data based on MongoDB. The web client sends a message through HTTP to the edge gateway and the edge gateway responds the result to the web client. In this process. The client support gateway bridges the web client and edge microservice to deliver the messages. In the second case, the client support gateway requests to the edge microservice (CG2EM) in the edge gateway. The IoT data is collected from IoT devices which send the data to the edge gateway periodically.

Figure 17 shows the experimental results of time delays for requesting the IoT data from the web client to the edge microservice of the edge gateway and requesting from the client support gateway to the edge microservice in the edge gateway. The result depicts C2EM takes more time than CG2EM because of the handling time by the client support gateway that takes time. The process in the edge gateway for the CG2EM, which presents the internal communication cost for accessing microservices in the same device.

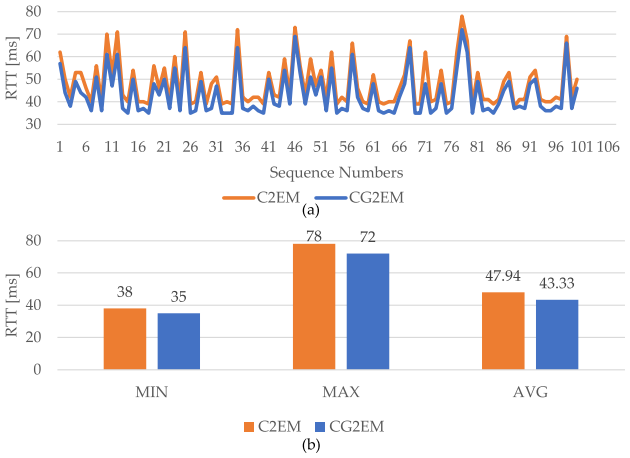


FIGURE 17. Experimental results of time delays for requesting IoT data.

For providing the evaluation of the memory usage on the edge gateway, the htop is used for monitoring the edge gateway based on the Raspberry Pi 4 with Ubuntu OS. As shown in Figure 18, the operating processes are assigned to use 1,626,252 kb from the total memory which is 3,885,009 kb. Therefore, the available memory is 2,258,757 kb to be used for more additional applications. For operating the modules of the edge gateway, the machine assigns the memory usage for 991,109 kb, and 635,143 kb is used for other applications and OS. The client support gateway and additional services

TABLE 1. Comparison of the main characteristics. The (Yes) means advantages and (No) means weaknesses.

Characteristics	IoTivity [50] (OCF) [47]	OCEAN [72] (oneM2M) [73]	IIC [33]	Global Edge [34]	Proposed Platform
Communication Variety	No	Yes	Yes	Yes	Yes
Device Management	No	Yes	Yes	Yes	Yes
Data Management	No	Yes	Yes	Yes	Yes
UI	No	Yes	Yes	No	Yes
REST API	Yes	Yes	Yes	Yes	Yes
Security	No	Yes	Yes	Yes	Yes
Microservice (Extendable)	Yes	No	No	No	Yes
Open Source	Yes	Yes	No	Yes	Yes

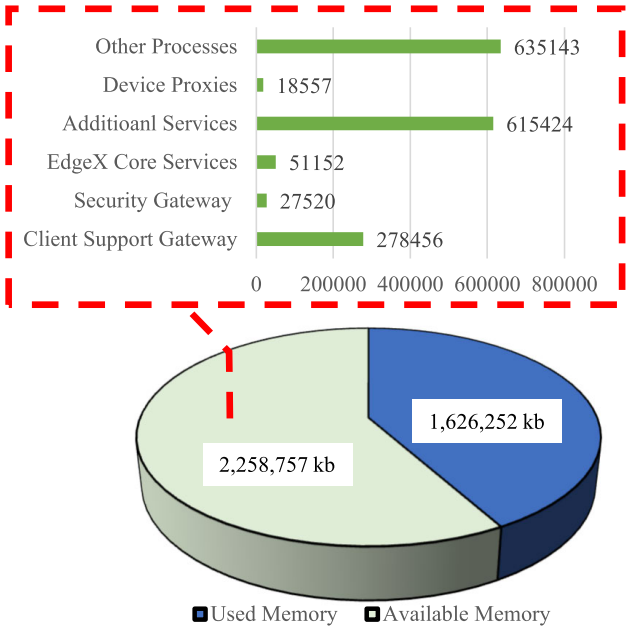


FIGURE 18. Experimental results of memory usage for edge gateway.

consume more memory usage because of the applications are developed in Java and operated through Java Virtual Machine (JVM). EdgeX core services, security gateway, and device proxies are developed in Go to provide services that are assigned 51,152 kb, 27,520 kb, and 18,557 kb. Therefore, the experiment presents Go-based microservices providers are assigned less memory although providing more APIs. The memory ability of the machine is sufficient for operating these processes including proposed edge gateway with security functions, edge computing management and device proxy.

VI. COMPARISON AND SIGNIFICANCE

In this section, we provide a comparison with the existing solutions for emphasizing the significance of the proposed secure edge computing management. The proposed edge computing management as a standalone integrated service provider that enables the function of interface, security and management in devices at the edge of the network. Most of IoT solutions interact with cloud servers to manage data. Therefore, the specifications are proposed to focus on communications such as OCF and LWM2M. However, management architectures such as oneM2M and OMA DM are recommended to deploy in a cloud level server for a large-scale deployment of devices. For decreasing network delays and distributing computing ability in private networks, the paradigm of edge computing inspires IoT platforms to deploy in edge of networks through the small size of devices.

Table 1 presents the advantages and weaknesses of the proposed edge computing architecture by comparing the main characteristics with other IoT and edge computing architectures which are introduced in the section of related works. For developing IoT devices to communicate through wireless and wired networks, heterogeneous communication protocols are proposed such as CoAP and MQTT. Moreover, based on those protocols or existing low level and application-level protocols, communication solutions including frameworks, libraries and protocols are proposed. However, satisfying the heterogeneity of communication for various devices in industries is difficult. Between different protocols, a proxy can be used for translating the messages for request and response. The proxy-based solution is popular to be used for bridging different protocols. Management of device and data is an important function that enables to discover the services of IoT devices. The management function is provided by a server through web services. REST API is the most used web service style to deliver data to the client-side. The server can provide a REST API for the management and UI services. According to the ability of the server, the functions of management and UI can be included. Moreover, based on REST API with multiple web services, security function is an important element to protect the data from the platform including repository and environment. On the server-side, a session can be used for remembering the client to identify the authenticated users. For REST APIs, the token-based authorization is a most-used solution such as OAuth and JWT. Microservices have become hugely popular in recent years because of scalability and greater agility in the era of containerization and cloud computing. Due to the nature of microservices, the applications of heterogeneous IoT devices are enabled to be built using different protocols, functions and interfaces based on a unified platform.

For comparing with the proposed edge computing platform, we select IoTivity, OCEAN, IIC and Global Edge (GE)

to depict characteristics. IoTivity is an open souse IoT framework that is the official implementation of OCF. The framework is developed based on CoAP to provide communication between devices. IoTivity focuses on communications between smart devices such as home appliances and personal mobile devices. Therefore, the management function is included in the OCF specification as well as in the implementation. However, CoAP is an application-level protocol that is designed for the REST architecture. Moreover, IoTivity is a light-weight framework that can be deployed as a microservice server to provide services. Different from OCF, oneM2M focuses on large scale applications such as factory, building, and city. The implementation is OCEAN that provides 2 parts which are Mobius and CUBE. For large scale device deployment, the management and its UI are provided. According to the oneM2M specification, the implementation requires the number of functions to provide communication and management for heterogeneous industrial products. Therefore, the fundamental implementation requires a high-performance machine to run Mobius. Accordingly, the microservices are provided to connect with other independent applications.

For deploying the computing systems to the edge of the network, IIC and GE propose reference architectures to develop the elements in the edge computing. IIC presents three layers including a layer of edge, platform and enterprise to provide edge computing services for a large-scale domain. In the enterprise layer, the management and UI functions are provided through interacting with devices of the edge layer. However, each of the characteristics can be changed according to the use case. The flexibility of the enterprise layer, that can run specific applications such as decision support systems, end-user interfaces, and operations management in the cloud. Most of the characteristics are involved in the enterprise layer. Therefore, once the reference architecture is deployed, then not easy to extend, unlike microservice providers through interacting with each other using microservices. Similar to IIC, GE also deploys the most functions on the business solution layer that runs in the cloud. However, the implementation of GE supports an open source license, unlike IIC.

The proposed edge computing platform tries to include all functions in the standalone server to deploy in an edge network. The microservice is the key to support the extendable applications that are integrated into the server. Therefore, the proposed edge computing is possible to involve management, UI, and security in the edge gateway based on EdgeX framework.

VII. CONCLUSION AND FUTURE DIRECTIONS

In this paper, we proposed the design and implementation of edge computing management based on secure microservices for IoT networks. The proposed edge computing management is provided by the edge gateway that is deployed in a local network to provide management of device, data, user, configuration and other additional functions through microservices for IoT networks. The proposed edge gateway is a hub that bridges web clients and IoT devices to communicate. Based on the EdgeX framework with multiple server modules, the edge gateway is enabled to scale up and down according to the capability of the device and standalone management service is provided through microservices. For supporting secure services from the edge gateway, a security gateway is deployed in the edge gateway to provide security functions based on authentication and authorization. Using the security gateway, the edge gateway allows the verified requests to the local network where the edge gateway and IoT devices are deployed. According to the experimental results, communications delays are very small for providing microservices based on the security gateway. Therefore, the proposed secure edge services can be considered to be used in real-time communications in a local network. Especially, offloading solutions for decision making, prediction, and classification based on interacting with secure microservices can be proposed.

As future directions, we will apply secure edge computing management to smart spaces to provide intelligent services based on trained prediction models. The models can be included in service providers to be invoked through microservices with parameters. The intelligent services will be developed based on microservices without revising the proposed secure edge computing management to update the edge gateway. Moreover, based on the extendable characteristic of microservices, we can develop the edge gateway to use the domain-specific data to extend the intelligent functions through microservices for serving heterogeneous industrial domains.

ACKNOWLEDGMENT

Any correspondence related to this article should be addressed to Dohyeun Kim.

CONFLICT OF INTERESTS

The authors declare that there is no conflict of interests regarding the publication of this article.

REFERENCES

[1] W. Jin and D. Kim, "Consistent registration and discovery scheme for devices and Web service providers based on RAML using embedded RD in OCF IoT network," *Sustainability*, vol. 10, no. 12, p. 4706, Dec. 2018.

[2] O. Salman, I. Elhajj, A. Kayssi, and A. Chehab, "Edge computing enabling the Internet of Things," in *Proc. IEEE 2nd World Forum Internet Things (WF-IoT)*, Dec. 2015, pp. 603–608.

[3] W. Shi, J. Cao, Q. Zhang, Y. Li, and L. Xu, "Edge computing: Vision and challenges," *IEEE Internet Things J.*, vol. 3, no. 5, pp. 637–646, Oct. 2016.

[4] A. Greenberg, J. Hamilton, D. A. Maltz, and P. Patel, "The cost of a cloud: Research problems in data center networks," 2008.

[5] E. Cuervo, A. Balasubramanian, D.-K. Cho, A. Wolman, S. Saroiu, R. Chandra, and P. Bahl, "MAUI: Making smartphones last longer with code offload," in *Proc. 8th Int. Conf. Mobile Syst., Appl., Services*, 2010, pp. 49–62.

[6] M. Satyanarayanan, P. Bahl, R. Caceres, and N. Davies, "The case for VM-based cloudlets in mobile computing," *IEEE Pervas. Comput.*, vol. 8, no. 4, pp. 14–23, Oct. 2009.

[7] F. Bonomi, R. Milito, J. Zhu, and S. Addepalli, "Fog computing and its role in the Internet of Things," in *Proc. 1st Ed MCC Workshop Mobile Cloud Comput.*, 2012, pp. 13–16.

[8] R. Morabito, V. Cozzolino, A. Y. Ding, N. Beijar, and J. Ott, "Consolidate IoT edge computing with lightweight virtualization," *IEEE Netw.*, vol. 32, no. 1, pp. 102–111, Jan. 2018.

[9] R. Morabito, R. Petrolo, V. Loscrì, and N. Mitton, "LEGIoT: A lightweight edge gateway for the Internet of Things," *Future Gener. Comput. Syst.*, vol. 81, pp. 1–15, Apr. 2018.

[10] C.-H. Chen, M.-Y. Lin, and C.-C. Liu, "Edge computing gateway of the industrial Internet of Things using multiple collaborative microcontrollers," *IEEE Netw.*, vol. 32, no. 1, pp. 24–32, Jan. 2018.

[11] R. Morabito, R. Petrolo, V. Loscrí, and N. Mitton, "Enabling a lightweight edge gateway-as-a-service for the Internet of Things," in *Proc. 7th Int. Conf. Netw. Future (NOF)*, Nov. 2016, pp. 1–5.

[12] W. Jin and D.-H. Kim, "IoT device management architecture based on proxy," in *Proc. 6th Int. Conf. Comput. Sci. Netw. Technol. (ICCSNT)*, Oct. 2017, pp. 84–87.

[13] W. Jin and D. Kim, "Resource management based on OCF for device self-registration and status detection in IoT networks," *Electronics*, vol. 8, no. 3, p. 311, Mar. 2019.

[14] W. Jin and D. Kim, "Development of virtual resource based IoT proxy for bridging heterogeneous Web services in IoT networks," *Sensors*, vol. 18, no. 6, p. 1721, May 2018.

[15] W. Jin and D. Kim, "Improved resource directory based on DNS name self-registration for device transparent access in heterogeneous IoT networks," *IEEE Access*, vol. 7, pp. 112859–112869, 2019.

[16] Z. Sheng, C. Mahapatra, C. Zhu, and V. C. M. Leung, "Recent advances in industrial wireless sensor networks toward efficient management in IoT," *IEEE Access*, vol. 3, pp. 622–637, 2015.

[17] J. Silva, J. Rodrigues, J. Al-Muhtadi, R. Rabêlo, and V. Furtado, "Management platforms and protocols for Internet of Things: A survey," *Sensors*, vol. 19, no. 3, p. 676, Feb. 2019.

[18] Y. Liu, S. Zhao, B. Cheng, and D. Zhang, "Heterogeneous sensors access middleware platform based on OSGI," in *Proc. 4th Int. Conf. Comput. Sci. Netw. Technol. (ICCSNT)*, vol. 1, Dec. 2015, pp. 447–451.

[19] C.-S. Shih, J.-J. Chou, N. Reijers, and T.-W. Kuo, "Designing CPS/IoT applications for smart buildings and cities," *IET Cyber.-Phys. Syst. Theory Appl.*, vol. 1, no. 1, pp. 3–12, Dec. 2016.

[20] M. A. Khan and K. Salah, "IoT security: Review, blockchain solutions, and open challenges," *Future Gener. Comput. Syst.*, vol. 82, pp. 395–411, May 2018.

[21] J. Zhang, B. Chen, Y. Zhao, X. Cheng, and F. Hu, "Data security and privacy-preserving in edge computing paradigm: Survey and open issues," *IEEE Access*, vol. 6, pp. 18209–18237, 2018.

[22] R. Xu, W. Jin, and D. Kim, "Microservice security agent based on API gateway in edge computing," *Sensors*, vol. 19, no. 22, p. 4905, Nov. 2019.

[23] *Edgex Foundry*. Accessed: Dec. 12, 2019. [Online]. Available: https://www.edgexfoundry.org

[24] *Kong*. Accessed: Dec. 12, 2019. [Online]. Available: https://konghq.com

[25] A. Ceselli, M. Premoli, and S. Secci, "Cloudlet network design optimization," in *Proc. IFIP Netw. Conf. (IFIP Netw.)*, May 2015, pp. 1–9.

[26] Z. Sanaei, S. Abolfazli, A. Gani, and R. Buyya, "Heterogeneity in mobile cloud computing: Taxonomy and open challenges," *IEEE Commun. Surveys Tuts.*, vol. 16, no. 1, pp. 369–392, 1st Quart., 2014.

[27] M. Satyanarayanan, R. Schuster, M. Ebling, G. Fettweis, H. Flinck, K. Joshi, and K. Sabnani, "An open ecosystem for mobile-cloud convergence," *IEEE Commun. Mag.*, vol. 53, no. 3, pp. 63–70, Mar. 2015.

[28] *Mobile Edge Computing-Introductory Technical White Paper*, document, etsi2014mobile, ETSI, 2014.

[29] Y. Mao, C. You, J. Zhang, K. Huang, and K. B. Letaief, "A survey on mobile edge computing: The communication perspective," *IEEE Commun. Surveys Tuts.*, vol. 19, no. 4, pp. 2322–2358, 2017.

[30] P. Mach and Z. Becvar, "Mobile edge computing: A survey on architecture and computation offloading," *IEEE Commun. Surveys Tuts.*, vol. 19, no. 3, pp. 1628–1656, 3rd Quart., 2017.

[31] K. Bilal, O. Khalid, A. Erbad, and S. U. Khan, "Potentials, trends, and prospects in edge technologies: Fog, cloudlet, mobile edge, and micro data centers," *Comput. Netw.*, vol. 130, pp. 94–120, Jan. 2018.

[32] S. Barbarossa, S. Sardellitti, and P. Di Lorenzo, "Communicating while computing: Distributed mobile cloud computing over 5G heterogeneous networks," *IEEE Signal Process. Mag.*, vol. 31, no. 6, pp. 45–55, Nov. 2014.

[33] M. Tseng, T. Canaran, and L. Canaran, "Introduction to edge computing in IIoT," Ind. Internet Consortium, Needham, MA, USA, White Paper, 2018.

[34] I. Sittón-Candanedo, R. S. Alonso, J. M. Corchado, S. Rodríguez-González, and R. Casado-Vara, "A review of edge computing reference architectures and a new global edge proposal," *Future Gener. Comput. Syst.*, vol. 99, pp. 278–294, Oct. 2019.

[35] A. Munir, P. Kansakar, and S. U. Khan, "IFCIoT: Integrated fog cloud IoT: A novel architectural paradigm for the future Internet of Things," *IEEE Consum. Electron. Mag.*, vol. 6, no. 3, pp. 74–82, Jul. 2017.

[36] M. Saad, "Fog computing and its role in the Internet of Things: Concept, security and privacy issues," *Int. J. Comput. Appl.*, vol. 180, no. 32, pp. 7–9, Apr. 2018.

[37] Y. Ai, M. Peng, and K. Zhang, "Edge computing technologies for Internet of Things: A primer," *Digit. Commun. Netw.*, vol. 4, no. 2, pp. 77–86, Apr. 2018.

[38] S. H. Mortazavi, M. Salehe, C. S. Gomes, C. Phillips, and E. de Lara, "Cloudpath: A multi-tier cloud computing framework," in *Proc. 2nd ACM/IEEE Symp. Edge Comput.*, Oct. 2017, pp. 1–13.

[39] F. Liu, G. Tang, Y. Li, Z. Cai, X. Zhang, and T. Zhou, "A survey on edge computing systems and tools," *Proc. IEEE*, vol. 107, no. 8, pp. 1537–1562, Aug. 2019.

[40] A. Musaddiq, Y. B. Zikria, O. Hahm, H. Yu, A. K. Bashir, and S. W. Kim, "A survey on resource management in IoT operating systems," *IEEE Access*, vol. 6, pp. 8459–8482, 2018.

[41] C. Legner and F. Thiesse, "RFID-based maintenance at frankfurt airport," *IEEE Pervas. Comput.*, vol. 5, no. 1, pp. 34–39, Feb. 2006.

[42] L. Tan and N. Wang, "Future Internet: The Internet of Things," in *Proc. 3rd Int. Conf. Adv. Comput. Theory Eng. (ICACTE)*, vol. 5, Aug. 2010, pp. V5-376–V5-380.

[43] S. Bhatia, A. Chauhan, and V. K. Nigam, "The Internet of Things: A survey on technology and trends," *Int. Res. J. Eng. Technol. (IRJET)*, vol. 3, no. 5, pp. 1397–1405, 2016.

[44] *OMA*. Accessed: Dec. 12, 2019. [Online]. Available: www.openmobilealliance.org

[45] C. A. L. Putera and F. J. Lin, "Incorporating OMA lightweight M2M protocol in IoT/M2M standard architecture," in *Proc. IEEE 2nd World Forum Internet Things (WF-IoT)*, Dec. 2015, pp. 559–564.

[46] S. Rao, D. Chendanda, C. Deshpande, and V. Lakkundi, "Implementing LWM2M in constrained IoT devices," in *Proc. IEEE Conf. Wireless Sensors (ICWiSe)*, Aug. 2015, pp. 52–57.

[47] *OCF*. Accessed: Dec. 12, 2019. [Online]. Available: https://openconnectivity.org/

[48] S. Park, "OCF: A new open IoT consortium," in *Proc. 31st Int. Conf. Adv. Inf. Netw. Appl. Workshops (WAINA)*, Mar. 2017, pp. 356–359.

[49] J.-C. Lee, J.-H. Jeon, and S.-H. Kim, "Design and implementation of healthcare resource model on IoTivity platform," in *Proc. Int. Conf. Inf. Commun. Technol. Converg. (ICTC)*, Oct. 2016, pp. 887–891.

[50] *Iotivity*. Accessed: Dec. 12, 2019. [Online]. Available: https://iotivity.org

[51] C. Akasiadis, V. Pitsilis, and C. D. Spyropoulos, "A multi-protocol IoT platform based on open-source frameworks," *Sensors*, vol. 19, no. 19, p. 4217, Sep. 2019.

[52] H. Park, H. Kim, H. Joo, and J. Song, "Recent advancements in the Internet-of-Things related standards: A oneM2M perspective," *ICT Express*, vol. 2, no. 3, pp. 126–129, Sep. 2016.

[53] M. A. Zamora-Izquierdo, J. Santa, J. A. Martínez, V. Martínez, and A. F. Skarmeta, "Smart farming IoT platform based on edge and cloud computing," *Biosyst. Eng.*, vol. 177, pp. 4–17, Jan. 2019.

[54] D. Mijić and E. Varga, "Unified IoT platform architecture platforms as major IoT building blocks," in *Proc. Int. Conf. Comput. Netw. Commun. (CoCoNet)*, Aug. 2018, pp. 6–13.

[55] N. Dragoni, S. Giallorenzo, A. L. Lafuente, M. Mazzara, F. Montesi, R. Mustafin, and L. Safina, "Microservices: Yesterday, today, and tomorrow," in *Present and Ulterior Software Engineering*. Cham, Switzerland: Springer, 2017, pp. 195–216.

[56] P. Di Francesco, P. Lago, and I. Malavolta, "Migrating towards microservice architectures: An industrial survey," in *Proc. IEEE Int. Conf. Softw. Archit. (ICSA)*, Apr. 2018, pp. 29–2909.

[57] S. Newman, *Building Microservices: Designing Fine-Grained Systems*, Newton, MA, USA, O'Reilly Media, 2015.

[58] M. Fowler and J. Lewis. (2014). *Microservices*. [Online]. Available: http://martinfowler.com/articles/microservices.html

[59] C. Santana, B. Alencar, and C. Prazeres, "Microservices: A mapping study for Internet of Things solutions," in *Proc. IEEE 17th Int. Symp. Netw. Comput. Appl. (NCA)*, Nov. 2018, pp. 1–4.

[60] A. Benayache, A. Bilami, S. Barkat, P. Lorenz, and H. Taleb, "MsM: A microservice middleware for smart WSN-based IoT application," *J. Netw. Comput. Appl.*, vol. 144, pp. 138–154, Oct. 2019.
[61] J.-M. Fernandez, I. Vidal, and F. Valera, "Enabling the orchestration of IoT slices through edge and cloud microservice platforms," *Sensors*, vol. 19, no. 13, p. 2980, Jul. 2019.
[62] K. Thramboulidis, D. C. Vachtsevanou, and A. Solanos, "Cyber-physical microservices: An IoT-based framework for manufacturing systems," in *Proc. IEEE Ind. Cyber-Physical Syst. (ICPS)*, May 2018, pp. 232–239.
[63] A. Krylovskiy, M. Jahn, and E. Patti, "Designing a smart city Internet of Things platform with microservice architecture," in *Proc. 3rd Int. Conf. Future Internet Things Cloud*, Aug. 2015, pp. 25–30.
[64] T. Vresk and I. Čavrak, "Architecture of an interoperable IoT platform based on microservices," in *Proc. 39th Int. Conv. Inf. Commun. Technol., Electron. Microelectron. (MIPRO)*, May 2016, pp. 1196–1201.
[65] D. Lu, D. Huang, A. Walenstein, and D. Medhi, "A secure microservice framework for IoT," in *Proc. IEEE Symp. Service-Oriented Syst. Eng. (SOSE)*, Apr. 2017, pp. 9–18.
[66] X. He and X. Yang, "Authentication and authorization of end user in microservice architecture," *J. Phys. Conf. Ser.*, vol. 910, Oct. 2017, Art. no. 012060.
[67] P. Solapurkar, "Building secure healthcare services using OAuth 2.0 and JSON Web token in IOT cloud scenario," in *Proc. 2nd Int. Conf. Contemp. Comput. Informat. (IC3I)*, Dec. 2016, pp. 99–104.
[68] J. R. Huerga, A. Kovalevych, R. Buchanan, D. Lee, C. Mooy, and X. Bruhiere, "Kong: Becoming a king of API gateways," Tech. Rep., 2018.
[69] B. Leiba, "OAuth Web authorization protocol," *IEEE Internet Comput.*, vol. 16, no. 1, pp. 74–77, Jan. 2012.
[70] S. Emerson, Y.-K. Choi, D.-Y. Hwang, K.-S. Kim, and K.-H. Kim, "An OAuth based authentication mechanism for IoT networks," in *Proc. Int. Conf. Inf. Commun. Technol. Converg. (ICTC)*, Oct. 2015, pp. 1072–1074.
[71] A. Bhawiyuga, M. Data, and A. Warda, "Architectural design of token based authentication of MQTT protocol in constrained IoT device," in *Proc. 11th Int. Conf. Telecommun. Syst. Services Appl. (TSSA)*, Oct. 2017, pp. 1–4.
[72] *Ocean*. Accessed: Dec. 12, 2019. [Online]. Available: http://www.iotocean.org
[73] *oneM2M*. Accessed: Dec. 12, 2019. [Online]. Available: http://www.onem2m.org

WENQUAN JIN received the B.S. degree in computer science from the Yanbian University of Science and Technology, China, in 2013, and the M.S. and Ph.D. degrees in computer engineering from Jeju National University, South Korea, in 2015 and 2019, respectively. He has been a Postdoctoral Researcher with the Big Data Research Center, Jeju National University, since March 2019. His research interests include integrating the IoT and edge computing with intelligent services for smart spaces. His studies standard specifications and communication protocols, such as OCF, OMA-DM, LWM2M, IoT-A, oneM2M, CoAP, HTTP, and MQTT. Also, he studies in the area of optimization and machine learning for the smart homes and energy optimization based on particle swarm optimization, deep neural networks, and recurrent neural networks. He is currently the Manager of a research project that is supported by the National Research Foundation of Korea. Moreover, he participates in multiple projects from national funds and institutes, such as ITCR, IITP, and ETRI.

RONGXU XU received the B.S. degree in computer science from the Yanbian University of Science and Technology, China, in 2014, and the M.S. degree in computer engineering from Konkuk University, South Korea, in 2017. He is currently pursuing the Ph.D. degree with the Department of Computer Engineering, Jeju National University, South Korea. His research interest includes edge computing.

TAEWAN YOU received the M.S. degree in computer science and engineering from Seoul National University (SNU), South Korea, in 2004, and the Ph.D. degree in computer engineering from SNU. He joined ETRI in 2011. He is currently a Senior Research Member. His research interests include future Internet, IPv6, mobility, and multi-homing technologies. His work is also closely related to standardization activities in IETF as well as ITU-T. He has published five ITU-T recommendations as an Editor and has participated in EU-FP7.

YONG-GEUN HONG received the B.S., M.S., and Ph.D. degrees in computer engineering from Kyoungpook National University, Daegu, South Korea. He is currently the Director of the Electronics and Telecommunications Research Institute, Daejeon, South Korea, where he is currently a Project Leader of the IoT Network and Intelligent Network Research and Development Projects. He is also working for the IoT related standardization at IETF and ITU-T. His research interests include the Internet protocol, the IoT, network intelligence, and edge computing.

DOHYEUN KIM received the B.S. degree in electronics engineering and the M.S. and Ph.D. degrees in information telecommunication from Kyungpook National University, South Korea, in 1988, 1990, and 2000, respectively. He joined the Agency of Defense Development (ADD) from Match 1990 to April 1995. Since 2004, he has been with Jeju National University, South Korea, where he is currently a Professor with the Department of Computer Engineering. From 2008 to 2009, he had been with the Queensland University of Technology, Australia, as a Visiting Researcher. His research interests include sensor networks, M2M/IoT, energy optimization and prediction, intelligent service, and mobile computing.